Computing Approximate Common Factors of Matrix Polynomials

Antonio Fazzi[a,*], Nicola Guglielmi[a], Ivan Markovsky[b]

[a] *Gran Sasso Science Institute (GSSI), Viale F. Crispi 7, 67100 L'Aquila, Italy*
[b] *Vrije Universiteit Brussel (VUB), Department ELEC, Pleinlaan 2, 1050 Brussels, Belgium*

Abstract

Computation of (approximate) polynomials common factors is an important problem in several fields of science, like control theory and signal processing. While the problem has been widely studied for scalar polynomials, the scientific literature in the framework of matrix polynomials seems to be limited to the problem of exact greatest common divisors computation. In this paper, we generalize two algorithms from scalar to matrix polynomials. The first one is fast and simple. The second one is more accurate but computationally more expensive. We test the performances of the two algorithms and observe similar behavior to the one in the scalar case. Finally we describe an application to multi input multi output linear time-invariant dynamical systems.

Keywords: Matrix polynomials, Approximate GCD, Subspace method, Matrix ODEs

1. Introduction

Polynomials common factors computation is an important problem in several scientific fields due to its applications [1]. In this paper we deal with common factors for matrix polynomials, which are matrices whose elements are polynomials, or equivalently polynomials with matrix coefficients. Readers not familiar with matrix polynomials can refer for example to [2, 3].

The computation of a Gratest Common Divisor (GCD) $C(\lambda)$ of two polynomial matrices $A(\lambda)$ and $B(\lambda)$ appears in several problems in multivariable control [4, 5, 6]. The problem has been studied by many authors and through different techniques. Some authors find the GCD as a combination of polynomials [7] or transform the block matrix $[A(\lambda)\ B(\lambda)]$ into $[C(\lambda)\ \ 0]$ [8]. Other methods use the generalized Sylvester matrix [9, 10].

Most of the works in the literature study the properties of the resultant for matrix polynomials, e.g. [9, 11, 12, 13], or deal with the computation of

*Corresponding author
Email addresses: antonio.fazzi@gssi.it (Antonio Fazzi), nicola.guglielmi@gssi.it (Nicola Guglielmi), imarkovs@vub.ac.be (Ivan Markovsky)

the exact GCD for matrix polynomials [6, 10, 14]. Anyway some applications (see Section 6) require the computation of approximate common factors, due to measurement noise or other perturbations on the data.

The Approximate GCD problem has been extensively studied for scalar polynomials; in the framework of multivariable control systems we deal with matrix polynomials and, up to our knowledge, there is no algorithm for Approximate GCD computation in the matrix case. The goal of this paper is to generalize the algorithms in [15] and in [16, 17] from scalar to matrix polynomials.

The structure of the paper is the following: in Section 2 we present issues related to the exact GCD computation in the matrix case, and the properties of the generalized resultant; in Section 3 we propose a generalization of the subspace method [15] while in Section 4 we present the generalized ODE-based method [16, 17], looking at the performances of the two algorithms in Section 5. Finally in Section 6 we look at applications in the framework of linear time-invariant systems.

Notation.

- $A(\lambda)$, $B(\lambda)$ are the (square) coprime data polynomials, $\hat{A}(\lambda), \hat{B}(\lambda)$ are the perturbed polynomials having a common factor, and we factorize them as $\hat{A} = C\bar{A}$, $\hat{B} = C\bar{B}$; C denotes the (monic) common factor;
- m is the dimension of the matrices A, B, n is the degree of the polynomials (we assume they have the same degree), d is the degree of the sought common factor;
- $\mathcal{S}$ denotes the set of structured Sylvester matrices, whose dimensions depend on the parameter ℓ (see Section 2.2), and $P_{\mathcal{S}}(\cdot)$ is the operator which project the argument onto the set $\mathcal{S}$;
- we denote by $\|\cdot\|_2$ and $\|\cdot\|_F$ the 2 norm and the Frobenius norm, respectively. The inner product for two matrices is the usual Frobenius inner product: $\langle A, B\rangle = \operatorname{tr}\big(A^\top B\big)$.

We assume in the following that both the matrices A, B are square in order to simplify the notation. Anyway the proposed algorithms work also when one of the two matrices is rectangular, as it is needed in the applications (see Section 6). As pointed out in Remark 1 we need only two matrices having the same number of rows (columns).

2. Exact GCD for matrix polynomials

We analyze in this section how to approach the (exact) GCD computation in the case of matrix polynomials, emphasizing the main differences with respect to the scalar case. The first difference arising when we consider matrices instead of scalars is the loss of commutativity. Henceforth we need to distinguish between the right and left divisors. In the following we focus on left divisors but right divisors have obvious counterparts.

Definition 1. *A greatest common left divisor of two polynomial matrices $A(\lambda)$ and $B(\lambda)$, having the same number of rows, is any polynomial matrix $C(\lambda)$ such that*

1. *$C(\lambda)$ is a common left divisor of $A(\lambda)$ and $B(\lambda)$, i.e.*
$$A(\lambda) = C(\lambda)\bar{A}(\lambda) \qquad B(\lambda) = C(\lambda)\bar{B}(\lambda) \tag{1}$$
for some polynomial matrices $\bar{A}(\lambda)$, $\bar{B}(\lambda)$;

2. *$C(\lambda)$ is multiple of any other common left divisor $C_1(\lambda)$, i.e. $C(\lambda) = C_1(\lambda)M(\lambda)$ for some polynomial matrix $M(\lambda)$.*

Remark 1. The definition of left (right) divisor is meaningful only in the case the two matrices have the same number of rows (columns). If we transpose the matrix polynomials, we can switch between left and right common factors.

2.1. The GCD is not unique

In the framework of scalar polynomials, two GCDs are equivalent up to a constant factor. A similar property holds for matrix polynomials: two GCDs are equivalent up to multiplication with unimodular matrices.

Definition 2. *Let $U(\lambda)$ be a square polynomial matrix of dimension g. Then $U(\lambda)$ is a unimodular polynomial matrix if there exists a $g \times g$ polynomial matrix $V(\lambda)$ such that $V(\lambda)U(\lambda) = I$. Equivalently, if* $\det(U(\lambda))$ *is a non-zero constant.*

Definition 3. *Given two matrices $C_1(\lambda)$ and $C_2(\lambda)$, they are equivalent if and only if there exist unimodular matrices $U(\lambda)$, $V(\lambda)$ such that $C_1(\lambda) = U(\lambda)C_2(\lambda)V(\lambda)$.*

At this point we need a way to recognize when a given matrix is unimodular. It is not difficult to check that the following *elementary* transformations applied to a matrix $C(\lambda)$ correspond to a multiplication with a suitable unimodular matrix:

1. interchange two columns: it is equivalent to the multiplication with a permutation matrix;

2. multiply a column by a nonzero constant: it is equivalent to multiplication with a constant diagonal matrix;

3. replace the i-th column $c_i(\lambda)$ by $c_i(\lambda) + \lambda^d c_j(\lambda)$: this is equivalent to the multiplication with a polynomial matrix equal to the identity except for the presence of λ^d in the position (j, i);

4. all the previous transformations can be applied to the rows and they correspond to a premultiplication with a suitable unimodular matrix.

The matrices associated to these transformations are called *elementary* unimodular matrices. A finite sequence of these elementary operations generates a unimodular matrix. The following theorem [18] characterizes the set of unimodular matrices.

Theorem 1. *$U(\lambda)$ is unimodular if and only if it is a finite product of elementary unimodular matrices.*

Remark 2. The set of equivalent GCDs, according to Definition 3 and Theorem 1, is big and sometimes it can be difficult to understand if two given matrix polynomials are equivalent even for small dimensions. In order to make this problem milder we restrict, in the following, to the case of monic GCDs (there is some loss of generality since we restrict to the polynomials whose leading coefficient is full rank).

2.2. Sylvester matrices for matrix polynomials

Let A and B be $m \times m$ matrix polynomials of degree n. Thus

$$A(\lambda) = A_0 + A_1\lambda + \cdots + A_n\lambda^n \text{ with } A_n \neq 0, \tag{2}$$

$$B(\lambda) = B_0 + B_1\lambda + \cdots + B_n\lambda^n \text{ with } B_n \neq 0. \tag{3}$$

Definition 4. *A square matrix polynomial $P(\lambda)$ is regular if $det(P(\lambda)) \neq 0$ identically.*

We assume $n > 0$, and that the leading coefficients A_n and B_n are invertible, so the matrices are regular. We say that $\lambda_0 \in \mathbb{C}$ is a common eigenvalue of A and B if there exist a vector $x_0 \neq 0$ such that $A(\lambda_0)x_0 = 0$ and $B(\lambda_0)x_0 = 0$. In this case we refer to x_0 as a common eigenvector of A and B at λ_0. For A and B to have a common eigenvalue at λ_0 it is necessary that $det(A(\lambda_0)) = 0$ and $det(B(\lambda_0)) = 0$, but the converse is not true.

A useful tool in testing polynomials coprimeness is the Sylvester resultant: its straightforward generalization to the matrix case is the following $2mn \times 2mn$ structured matrix

$$\mathbf{S}(A,B) = \begin{pmatrix} A_n & \cdots & \cdots & A_0 & & & \\ & A_n & \cdots & \cdots & A_0 & & \\ & & \ddots & & & \ddots & \\ & & & A_n & \cdots & \cdots & A_0 \\ B_n & \cdots & \cdots & B_0 & & & \\ & B_n & \cdots & \cdots & B_0 & & \\ & & \ddots & & & \ddots & \\ & & & B_n & \cdots & \cdots & B_0 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} n \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} n \end{matrix} . \tag{4}$$

In [11] it has been shown that the key property for the classical Sylvester resultant does not carry over for matrix polynomials, in particular

$$\dim\ \ker(S(A,B)) \geq \nu(A,B), \tag{5}$$

where $\nu(A,B)$ denotes the total common multiplicity of the common eigenvalues of A and B. The following simple example shows that the inequality (5) can be strict:

$$A(\lambda)=\begin{pmatrix}-1+\lambda & 0\\ 1 & -1+\lambda\end{pmatrix},\quad B(\lambda)=\begin{pmatrix}\lambda & 1\\ 0 & \lambda-2\end{pmatrix}$$
$$\mathbf{S}(A,B)\begin{pmatrix}1\\-2\\1\\-1\end{pmatrix}=\begin{pmatrix}1&0&-1&0\\0&1&1&-1\\1&0&0&1\\0&1&0&-2\end{pmatrix}\begin{pmatrix}1\\-2\\1\\-1\end{pmatrix}=0, \tag{6}$$

so the kernel of the resultant has dimension 1, but $det(A(\lambda))$ and $det(B(\lambda))$ have no common zeros, hence the matrices have no common eigenvalues.

In order to get the equality in (5) we can consider a bigger Sylvester matrix [11]. Defining the following resultant

$$S_\ell(A,B)=\begin{pmatrix}A_n & \cdots & \cdots & A_0 & & & \\ & A_n & \cdots & \cdots & A_0 & & \\ & & \ddots & & & \ddots & \\ & & & A_n & \cdots & \cdots & A_0\\ B_n & \cdots & \cdots & B_0 & & & \\ & B_n & \cdots & \cdots & B_0 & & \\ & & \ddots & & & \ddots & \\ & & & B_n & \cdots & \cdots & B_0\end{pmatrix}\begin{matrix}\left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\}\ell-n\\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\}\ell-n\end{matrix} \tag{7}$$

we have the equality in (5) if $\ell \geq n(m+1)$. This can be useful if we want similar properties to the case of scalar polynomials. Going back to the polynomials in (6), we can check that

$$S_3(A,B)=\begin{pmatrix}1&0&-1&0&0&0\\0&1&1&-1&0&0\\0&0&1&0&-1&0\\0&0&0&1&1&-1\\1&0&0&1&0&0\\0&1&0&-2&0&0\\0&0&1&0&0&1\\0&0&0&1&0&-2\end{pmatrix}$$

is full rank.

2.3. The Approximate GCD problem

In the past years several authors have proposed some algorithms for the computation of an exact GCD of matrix polynomials. But in practical applications, the coefficients can be inexact due to several sources of error. Given coprime matrix polynomials, we are interested in computing the smallest perturbation which makes them no more coprime.

Consider two coprime matrix polynomials $A(\lambda)$ and $B(\lambda)$. The problem is to compute a closest pair of matrix polynomials $\hat{A}(\lambda)$, $\hat{B}(\lambda)$ which has a non trivial common factor of specified degree d. In the following we assume that the coefficient matrices are real. The distance between two pairs of matrix polynomials is defined as follows:

$$dist(\{A, B\}, \{\hat{A}, \hat{B}\}) = \sqrt{\sum_{j=0}^{n} \|A_j - \hat{A}_j\|_F^2 + \sum_{j=0}^{n} \|B_j - \hat{B}_j\|_F^2} \tag{8}$$

where A_j and B_j denote the j-th (matrix) coefficient of the corresponding matrix polynomials. The Approximate GCD problem is the following:

Problem 1. *Given two coprime matrix polynomials A and B, a number $d \in \mathbf{N}$, compute*

$$\inf_{\{\hat{A},\hat{B}\}:\hat{A},\hat{B} \text{ have a GCD of degree } d} dist(\{A, B\}, \{\hat{A}, \hat{B}\})$$

where the distance is the one defined in (8).

In the following sections we propose two algorithms for solving the nonconvex optimization Problem 1 by local optimization approaches. To the best of our knowledge there is no algorithm in the scientific literature to compute its solution. Our proposals come from the generalization of two algorithms from scalar to matrix polynomials: the subspace method [15] and an ODE-based algorithm [17]. We list for each algorithm the main points and properties, and we test their performance on some numerical examples.

3. Generalized Subspace method

In this section we describe how we generalize the subspace method [15] to the computation of an approximate GCD for matrix polynomials. The original algorithm for scalar polynomials is a powerful tool in the framework of GCD computation since it is *simple to develop, easy to understand and convenient to implement.* Moreover it is one of the first algorithms capable of dealing with noise-corrupted data. However, as shown in [17], the performance of the subspace method can be improved in terms of accuracy of the solution by other optimization methods. The basic idea of the algorithm is the fact that the informations on the (approximate) GCD of a set of polynomials can be extracted from the left null space of the associated resultant.

We briefly recall how the algorithm works for scalar polynomials:

1. Build S, the Sylvester matrix of dimension $N(n+1) \times (2n+1)$ associated to the given data polynomials, where N is the number of polynomials and n is the degree of the polynomials (we consider the maximum degree if the polynomials have different degrees).

2. a Compute
$$U_0 = (U_{r+1}, \dots, U_{2n+1}),$$
the left null space of $S^\top$, where $r = 2n+1-d$ (d is the degree of the GCD). U_0 has d columns.

   b In order to extract the information about the GCD, reshape each column of U_0 into an Hankel matrix with r rows:
$$H_i = \begin{pmatrix} U_i(1) & U_i(2) & \cdots & U_i(d+1) \\ U_i(2) & U_i(3) & \cdots & U_i(d+2) \\ \cdots & \cdots & \cdots & \cdots \\ U_i(r) & U_i(r+1) & \cdots & U_i(2n+1) \end{pmatrix} \qquad i = r+1, \dots, 2n+1. \tag{9}$$

3. Build the matrix
$$R = \sum_{i=r+1}^{2n+1} H_i^T H_i$$
and extract the GCD by the eigenvector of R corresponding to the smallest eigenvalue.

To generalize the method for matrix polynomials, we replace the scalar coefficients by matrices of dimension m, manipulating and reshaping the data in a suitable way.

1. Build the structured Sylvester matrix S as in (4) of dimension $Nm(n+1) \times m(2n+1)$.

2. Compute
$$U_0 = (U_{r+1}, \dots, U_{2n+1}),$$
the left null space of $S^\top$, which has $k = md$ columns (the degree of the common factor times the dimension of the coefficient); we have now $r = m(2n+1-d)$, using the same notation r just to have a similar expression.

3. Reshape each vector of U_0 into a $m \times (2n+1)$ matrix (according to the dimension of the coefficients)
$$\bar{U}_i = \begin{pmatrix} U_i(1) & U_i(m+1) & \cdots & U_i(2mn+1) \\ \vdots & \vdots & & \vdots \\ U_i(m) & U_i(2m) & \cdots & U_i(m(2n+1)) \end{pmatrix} \text{for } i = r+1, \dots 2n+1$$
and build a $m(d+1) \times (2n+1-d)$ block Hankel matrix from the columns

of $\bar{U}_i$ for all $i = r+1, \dots, 2n+1$;

$$H_i = \left(\begin{array}{cccc}
U_i(1) & U_i(m+1) & \cdots & U_i((2n-d)m+1) \\
\vdots & \vdots & & \vdots \\
U_i(m) & U_i(2m) & \cdots & U_i((2n-d)m+m) \\
\hline
U_i(m+1) & \cdots & U_i((2n-d)m+1) & U_i((2n-d+1)m+1) \\
\vdots & & \vdots & \vdots \\
U_i(2m) & \cdots & U_i((2n-d)m+m) & U_i(2n-d+2)m \\
\hline
\vdots & & \vdots & \vdots \\
\hline
U_i(dm+1) & \cdots & U_i((2n-1)m+1) & U_i(2nm+1) \\
 & & \vdots & \\
U_i(m(d+1)) & \cdots & U_i(2nm) & U_i(m(2n+1))
\end{array}\right).$$

Stack then the matrices H_i in a row $\mathcal{K} = [H_{r+1}, \dots, H_{2d+1}]$.

4. The m left singular vectors of $\mathcal{K}$ associated to the m smallest singular values give the (approximate) common factor.

Remark 3. Given the matrices A and B, the subspace method computes only their GCD but not the perturbed polynomials $\hat{A}, \hat{B}$. To compute these polynomials we need to solve the least squares problem

$$\min_{\hat{A},\hat{B}} \|A - \hat{A}\|_2^2 + \|B - \hat{B}\|_2^2 = \min_{\bar{A},\bar{B}} \|A - C\bar{A}\|_2^2 + \|B - C\bar{B}\|_2^2 \tag{10}$$

where C is the common factor computed by the algorithm.

4. Generalized ODE method

The goal of this section is to generalize the algorithm proposed in [17] for scalar polynomials, to the case of matrix polynomials.

4.1. General aspects

We describe first some useful tools and ideas to understand how the algorithm works. When we deal with coprimeness of matrix polynomials, just as it happens for the scalar case, the Sylvester resultant is a useful tool. We showed in Section 2.2 that, replacing the scalar coefficients by matrices, we do not have anymore the equality between the co-rank of the resultant and the degree of the common factor between the polynomials, as it happens in the scalar case [19]. In order to solve this issue, it can be worth to work with the modified resultant S_ℓ (7), since in this way we preserve the equality in (5).

We start with the full rank Sylvester matrix $S_\ell(A, B)$ and we want to perturb the coefficients of the polynomials (in a minimal way) so that the kernel of the associated resultant $S_\ell(\hat{A}, \hat{B})$ has dimension $k = md$. This is done by iteratively adding a structured perturbation to the matrix $S_\ell(A, B)$ which minimizes the singular values of interest (the k smallest singular values, assuming they are increasing). The rank test on the Sylvester matrix is done by computing its SVD, and in particular it is well known that a matrix has co-rank k if and only if it has k zero singular values. Exploiting the fact that the singular values are ordered non negative real numbers, we can focus on minimizing only the k-th singular value. In particular we write the perturbed matrix as $\hat{S} = S_\ell + \epsilon E$, where ϵ is a scalar measuring the norm of the perturbation, while E is a norm one matrix (w.r.t. the Frobenius norm) which identifies as εE the minimizer of σ_k over the ball of matrices whose norm is at most ε. In this way we can move E and ϵ separately, minimizing the k-th singular value at one step, and the norm of the perturbation at the other until $\sigma_k = 0$.

These ideas give raise to the following 2-levels algorithm (the reader can check [16, 17] for further details in the case of scalar polynomials): we iteratively consider a matrix of the form $S_\ell + \epsilon E$ and we update it on two different levels:

- at the inner level we fix the value of ϵ, and we minimize the functional σ_k by looking for the stationary points of a system of ordinary differential equations for the matrix E;
- at the outer level, we move the value of ϵ in order to compute the best possible solution.

Remark 4. From the numerical point of view the functional σ_k cannot vanish, but it can only reach a fixed small tolerance.

4.2. The inner iteration

We analyze now the inner iteration of the algorithm, where the value of ϵ is fixed. The goal is to compute an optimal perturbation E that minimizes the singular value σ_k of the matrix $S_\ell + \epsilon E$ over the set of matrices E of unit Frobenius norm. To do this we consider a smooth path of matrices $E(t)$ of unit Frobenius norm along which the singular value σ_k of $S_\ell + \varepsilon E(t)$ decreases. We exploit the following result about derivatives of eigenvalues for symmetric matrices [20].

Lemma 1. *Let $D(t)$ be a differentiable real symmetric matrix function for t in a neighborhood of 0, and let $\lambda(t)$ be an eigenvalue of $D(t)$ converging to a simple eigenvalue λ_0 of $D(0)$ as $t \to 0$. Let x_0 be a normalized eigenvector of D_0 associated to λ_0. Then the function $\lambda(t)$ is differentiable near $t = 0$ with*

$$\dot{\lambda} = x_0^\top \dot{D} x_0 \tag{11}$$

Assuming that $E(t)$ is smooth we can apply Lemma 1 to the eigenvalues of the matrix $\hat{S}^\top(t)\hat{S}(t) = (S_\ell + \epsilon E(t))^\top (S_\ell + \epsilon E(t))$, and we observe that the eigenvalues of $\hat{S}^\top \hat{S}$ are the squares of the singular values of $\hat{S}$ (we can assume the singular values are differentiable functions since from the numerically point of view we do not observe any coalescence among them). Omitting the time dependence, we find the following expression for the derivative of σ_k:

$$\dot{\sigma}_k = \epsilon u^\top \dot{E} v, \tag{12}$$

where u, v are the singular vectors of $\hat{S}$ associated to σ_k; so the steepest descent direction for the functional σ_k, minimizing the function over the admissible set for $\dot{E}$, is attained by minimizing $u^\top \dot{E} v = \langle uv^\top, \dot{E}\rangle$ (remember $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product). We notice that $E \in \mathcal{S}$, and consequently $\dot{E} \in \mathcal{S}$, hence

$$\langle uv^\top, \dot{E}\rangle = \langle P_{\mathcal{S}}(uv^\top), \dot{E}\rangle \tag{13}$$

where the formula for the operator $P_{\mathcal{S}}$ (the projection of the argument onto the Sylvester structure) is given in the following Lemma (see [16, Lemma 4.2] for a proof in the scalar case):

Lemma 2. *Let S_ℓ be the set of generalized Sylvester matrices of dimension $m\ell \times 2m(\ell - n)$, and let $H \in \mathbb{R}^{m\ell \times 2m(\ell-n)}$ be an arbitrary matrix. The orthogonal projection with respect to the Frobenius norm of H onto $\mathcal{S}$ is given by (using Matlab notation for the rows/columns of the matrices)*

$$P_{\mathcal{S}}(H) = S_\ell(P^1, P^2),$$

where

$$P^1_{n-i} = \frac{1}{\ell - n} \sum_{j=1}^{\ell - n} H(m(j-1)+1 : mj, m(j-1)+1+mi : m(j+i))$$

$$P^2_{n-i} = \frac{1}{\ell - n} \sum_{j=1}^{\ell - n} H(m(\ell-n)+m(j-1)+1 : m(\ell-n)+mj, \dots$$

$$m(j-1)+1+mi : m(j+i))$$

for $\quad i = 0, \dots, n.$

We underline that the projection $P_{\mathcal{S}}(uv^\top)$ is different from zero for any pair of singular vectors u, v associated to a non-zero singular value (the reader can check [17, Lemma 3.3] for the proof).

4.2.1. Minimization problem

We found in (12) the expression for the derivative of the singular value σ_k of the Sylvester matrix $\hat{S} = S_\ell + \epsilon E$. In order to compute the steepest descent direction for σ_k we need to compute

$$G = arg \min_{\substack{\dot{E} \in \mathcal{S} \\ \|\dot{E}\| = 1 \\ \langle E, \dot{E}\rangle = 1}} u^\top \dot{E} v \tag{14}$$

where the constraint on the norm is added in order to select a unique solution, since we look for a direction. The solution of (14) is given by:

$$\dot{E} = -P_{\mathcal{S}}(uv^\top) + \langle E, P_{\mathcal{S}}(uv^\top)\rangle E \tag{15}$$

(check [16, Section 4.2] for the proof). Consequantly (15) is the key point of the inner iteration of the proposed algorithm. The following result shows its importance:

Theorem 2. *Let $E(t) \in \mathcal{S}$ be a matrix of unit Frobenius norm, which is a solution of* (15)*. If σ is the singular value of $\hat{S} = S_\ell + \epsilon E$ associated to the singular vectors u, v, then $\sigma(t)$ is decreasing, i.e.*

$$\dot{\sigma} \leq 0$$

Proof. In the proof we show that $\dot{\sigma} \leq 0$. We remember the expression for $\dot{\sigma} = u^\top \dot{E} v$ (up to constant factors). Exploiting the equation (15) to replace $\dot{E}$ we have two terms: the first is

$$u^\top P_{\mathcal{S}}(uv^\top)v = \langle uv^\top, P_{\mathcal{S}}(uv^\top)\rangle = \|P_{\mathcal{S}}(uv^\top)\|_F^2$$

since $P_{\mathcal{S}}(uv^\top)$ has the Sylvester structure, so the inner product does not change if we project the term $uv^\top$. The second is

$$u^\top \langle E, P_{\mathcal{S}}(uv^\top)\rangle E v = \langle E, P_{\mathcal{S}}(uv^\top)\rangle \langle E, uv^\top\rangle = \langle E, P_{\mathcal{S}}(uv^\top)\rangle^2$$

since E is a Sylvester matrix. Summing the two terms with the correct signs we have

$$\dot{\sigma} = u^\top \dot{E} v = -\|P_{\mathcal{S}}(uv^\top)\|_F^2 + \langle E, P_{\mathcal{S}}(uv^\top)\rangle^2 \leq 0$$

since $\|E\|_F = 1$. □

By Theorem 2 we have that stationary points of the ODE are candidate local minima for the functional under the considered constraints. The following corollary provides a rigorous characterization of minimizers.

Corollary 1. *Consider a solution of equation* (15)*, and assume the corresponding singular value $\sigma > 0$. The following statements are equivalent:*

1. $\dot{\sigma} = 0$
2. $\dot{E} = 0$
3. *E is a scalar multiple of $P_{\mathcal{S}}(uv^\top)$.*

4.2.2. The integration of the equation

We discuss here how to compute the solution of the ODE (15). Since (15) is a constrained gradient system, the value of σ_k is monotonically decreasing, as we can see in Figure 1. The function evaluation required in the integration of the equation is expensive since it involves the computation of a SVD decomposition

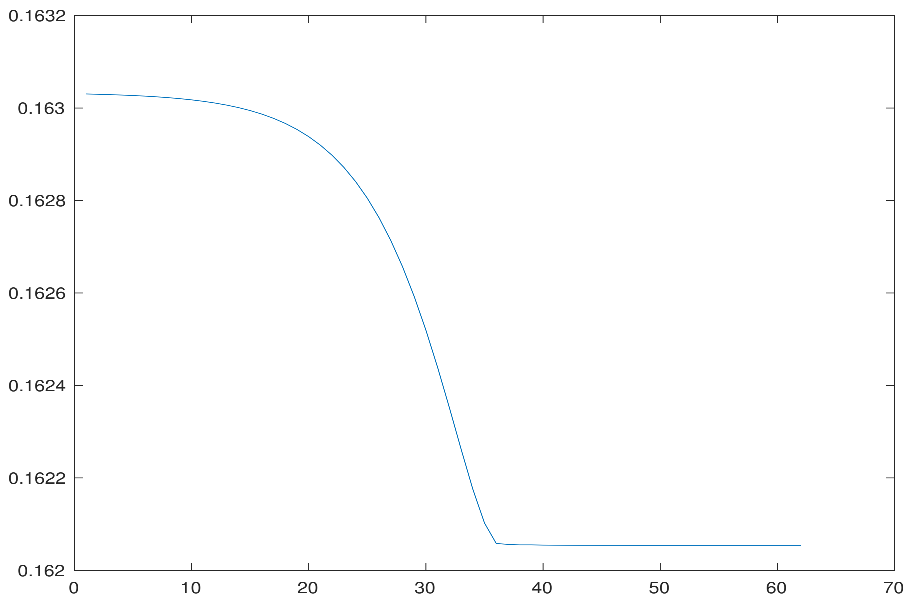

Figure 1: Tipical behavior of the singular value σ_k of the matrix $S_\ell + \epsilon E$ during the integration of the ODE (15) for a fixed ϵ. The final value corresponds to the stationary point of the equation (15).The data for the plot are choosen randomly.

at each step (we need both the singular value and the corresponding singular vectors), so a suitable choice is that of using the explicit Euler scheme. We summarize the pseudo-code in Algorithm 1. We remark that the performances of the code can change depending on the values of some parameters (which depend on the starting data).

Algorithm 1: Numerical solution of the ODE (15)

```
Data: $A, B$ (or equivalently the associated Sylvester matrix), $\sigma_k$, $u$, $v$, $h$
      (step Euler method), $\gamma$ (step size reduction), tol and $\epsilon$.
Result: $\bar{E}$, $\bar{\sigma}_k$, $\bar{u}$, $\bar{v}$ and $\bar{h}$
begin
1   Set $\tilde{h} = h$
2   Compute $\dot{E} = -P_S(uv^\top) + \langle E, P_S(uv^\top)\rangle E$
3   Euler step $\rightarrow \tilde{E} = E + h\dot{E}$
4   Normalize $\tilde{E}$ dividing it by its Frobenius norm
5   Compute the singular value $\tilde{\sigma}_k$ of the matrix $\tilde{S} = S_\ell + \epsilon\tilde{E}$
6   Compute the singular vectors $\tilde{u}$ and $\tilde{v}$ of the matrix $\tilde{S}$ associated to
    $\tilde{\sigma}_k$
7   if $\tilde{\sigma}_k > \sigma_k$ then
        reject the result and reduce the step $\tilde{h}$ by a factor $\gamma$
        repeat from 3
    else
        accept the result; set $\bar{h} = h$, $\bar{\sigma}_k = \tilde{\sigma}_k$, $\bar{u} = \tilde{u}$, $\bar{v} = \tilde{v}$
8       if $\bar{\sigma}_k - \sigma_k < tol$ or $\bar{\sigma}_k \leq tol$ then
            return
9   if $\bar{h} = h$ then
        increase the step size of $\gamma$, $\bar{h} = \gamma h$
    else
        set $\bar{h} = h$
10  Go to the next iteration
```

4.3. The outer iteration

After integrating the ODE (15) till a stationary point we know the value σ_k (for a fixed value of ϵ) which we denote as $\sigma_k(\epsilon)$, and we need to find the minimal value ε (that is the norm of the perturbation to the original Sylvester matrix) which solves the problem $\sigma_k(\varepsilon) = 0$. Observe that $\hat{S} - S_\ell = \varepsilon E$, hence the distance between the two matrices (and consequently the distance between the matrix polynomials defined in (8)) is related to the value of ε.

As long as we are looking for the zero of a function, we can use a root finding algorithm, and for this purpose it can be useful the following Lemma which gives the expression of the derivative of σ_k with respect to ϵ.

Lemma 3. *[17] Assume that*

1. *$\epsilon \in (0, \bar{\epsilon})$ (for a certain upper bound $\bar{\epsilon}$) such that $\sigma_k(\epsilon) \neq 0$ for the simple singular value σ_k;*

2. *$\sigma_k(\epsilon)$ and $E(\epsilon)$ are smooth functions with respect to ϵ.*

If $u(\varepsilon)$ and $v(\varepsilon)$ are the singular vectors of $\hat{S} = S_\ell + \varepsilon E$ associated to σ_k, then

$$\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\sigma_k(\epsilon) = -\|P_{\mathcal{S}}(u(\varepsilon)v(\varepsilon)^\top)\|_F.$$

Figure 2 shows the typical behavior of the function $\sigma_k(\varepsilon)$. Thanks to Lemma

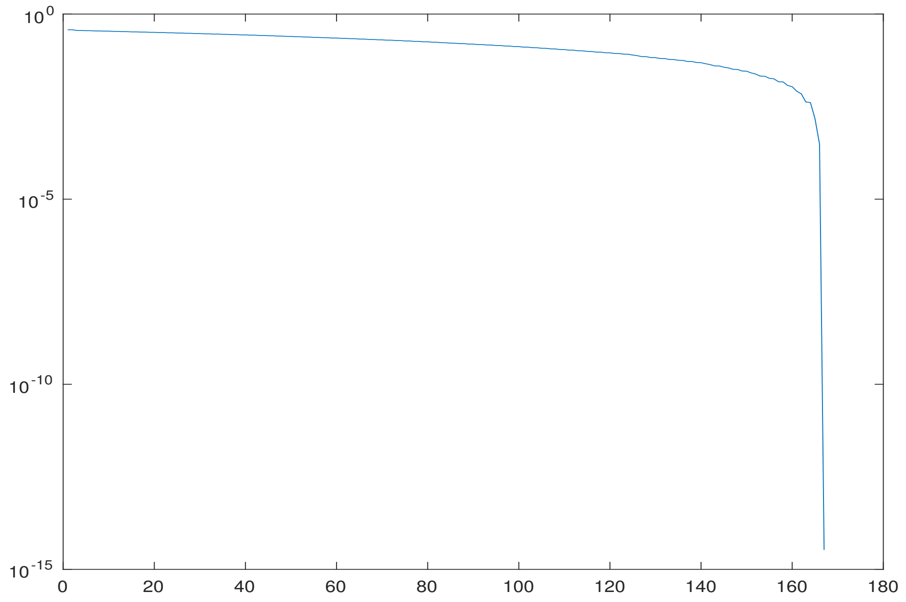

Figure 2: Plot of the function $\sigma_k(\epsilon)$ for a random example.

3 we can choose the Newton's method as a root finding algorithm, coupled with a bisection step in order to look for a better solution once the singular value reaches the value zero (or the fixed tolerance from a numerical point of view). We summarize the algorithm for the outer iteration in Algorithm 2.

Algorithm 2: Computation of ϵ^*

```
   Data: $tol$, $tol_2$ and $\epsilon_{\min}$. $\epsilon_{\max}$ lower and upper bounds, $\epsilon$, $\sigma(\epsilon)$ current
         values
   Result: $\epsilon_f$ approximation of $\epsilon^*$
   begin
1      Set Bisect = False, $j = 1$, $\varepsilon_j = \varepsilon$
2      while $|\epsilon_j - \epsilon_{\max}| \geq \text{tol}_2$ do
3          if $Bisect = False$ then
               Store $\epsilon_t$ and $\sigma(\epsilon_t)$
4              Compute $\tilde{\epsilon}_{j+1}$ by a Newton iteration
5              if $\tilde{\epsilon}_{j+1} \notin (\epsilon_{\min}, \epsilon_{\max})$ then
                   Set $\tilde{\epsilon}_{j+1} = (\epsilon_{\max} + \epsilon_{\min})/2$
           else
               Set $\tilde{\epsilon}_{j+1} = (\epsilon_{\max} + \epsilon_{\min})/2$

6          Compute $\sigma(\tilde{\epsilon}_{j+1})$ by integrating the ODE (Algorithm 1)
7          if $\sigma(\tilde{\epsilon}_{j+1}) < \text{tol}$ then
               Set Bisect = True
8              Set $\epsilon_{\max} = \tilde{\epsilon}_{j+1}$
           else
               Set Bisect = False
9              $\varepsilon_{min} = \tilde{\epsilon}_{j+1}$
10             Set $\epsilon_{j+1} = \tilde{\epsilon}_{j+1}$
11             Set $j = j + 1$

12     Set $\epsilon_f = \epsilon_j$
```

Observe the presence of two different tolerances: tol is related to the threshold of the singular value, while tol_2 identifies the interval containing ϵ^*.

Remark 5. Increasing the value of ε, due to the choice of an initial value for the matrix E in the ODE (15), it could happen that the computed $\sigma_k(\varepsilon)$ does not decrease. In order to have a global decreasing property with respect to t and ε we can replace Algorithm 1 by alternating the following dynamics:

1. consider the matrix $S_\ell + \epsilon E$ computed at the last step, and integrate the equation (15) removing any constraint on the norm of E, till the norm of the perturbation reaches a fixed level ϵ_2;

2. starting from the solution computed in point 1, integrate the equation adding the constraint $\|E\| = \varepsilon_2$ (using Algorithm 1).

4.4. How to compute the GCD

In this paragraph we discuss how to extract the GCD from the perturbed polynomials computed by the ODE-based algorithm proposed in Section 4. We saw (Remark 3) that given the GCD, we can obtain the polynomials $\hat{A}$, $\hat{B}$ by solving a least squares problem, but here the problem is more difficult.

The first idea to compute the sought common factor from the non-coprime polynomials $\hat{A}, \hat{B}$ is to apply a fast and computationally cheap algorithm (e.g. the subspace method proposed in Section 3).

Alternatively we can make use of an external function for (exact) GCD computation for matrix polynomials. A suitable function comes from the Polyx Toolbox (`www.polyx.com`), referring to the function *grd.m* or *gld.m* depending on the interest in computing a right or a left common factor, respectively.

The functions grd.m and gld.m. We briefly explain here how the two functions *grd.m* and *gld.m* from the Polyx Toolbox (`www.polyx.com`) work. We state the idea of the algorithm for right common factors computation, but dealing with left common factors has analogous counterparts.

Consider the matrix polynomials

$$\begin{aligned} N_1(z) &= N_{10} + N_{11} z + \cdots + N_{1w} z^w \\ N_2(z) &= N_{20} + N_{21} z + \cdots + N_{2w} z^w \end{aligned}$$

having the same number of columns m_N, and define $N = \begin{bmatrix} N_1 \\ N_2 \end{bmatrix}$. Consider the resultants $S_{w+\ell}(N_1, N_2)$ (as defined in (7)) for increasing $\ell = 1, 2, \ldots$. Each Sylvester matrix is then reduced to its shifted row Echelon form by a Gaussian elimination algorithm without row permutations. According to [21] the last m_N nonzero rows of $S_{\bar{\ell}}$ yield the coefficients of a greatest common right divisor of N_1, N_2, where $\bar{\ell}$ is defined as the smallest integer such that

$$\operatorname{rank}(S_{w+\bar{\ell}+1}) - \operatorname{rank}(S_{w+\bar{\ell}}) = m_N.$$

However we remember these functions are thought for exact GCD computation, while the output polynomials computed by the proposed ODE based algorithm have not an exact GCD (the singular values of the resultant decrease up to a small tolerance but they do not reach the zero). In particular, we can observe some of the following issues:

1. the computed GCD equals the identity (so the functions are not able to reveal the presence of a common factor);
2. the leading coefficient of the GCD is singular, while we always assume the common factors are monic (in particular the leading coefficient is full rank);
3. the computed GCD has degree higher than expected.

Most of the times no one of the previous facts is verified, and in these cases the common factors computed by the function *grd.m* match the ones computed by the subspace method.

5. Numerical experiments

We look now at some numerical experiments in order to understand the performances of the proposed algorithms. As stated before, there is no term of comparison in the scientific literature (up to our knowledge), so the results of our algorithms are compared with the solutions obtained through the Matlab function $fminsearch$. We consider random generated data having an exact common factor, and we add different levels of perturbations (in the interval $[0, 1]$) in order to analyze the solution computed by the different approaches. We neglect the numerical values of the polynomials, and we focus on the values of the computed distances. In the following experiments we plot the distance averaged over fifty runs of the algorithms with different noise realizations.

In the considered example (see Figure 3) we have two 2×2 matrix polynomials of degree 3 and we compute an approximate common factor of degree one.

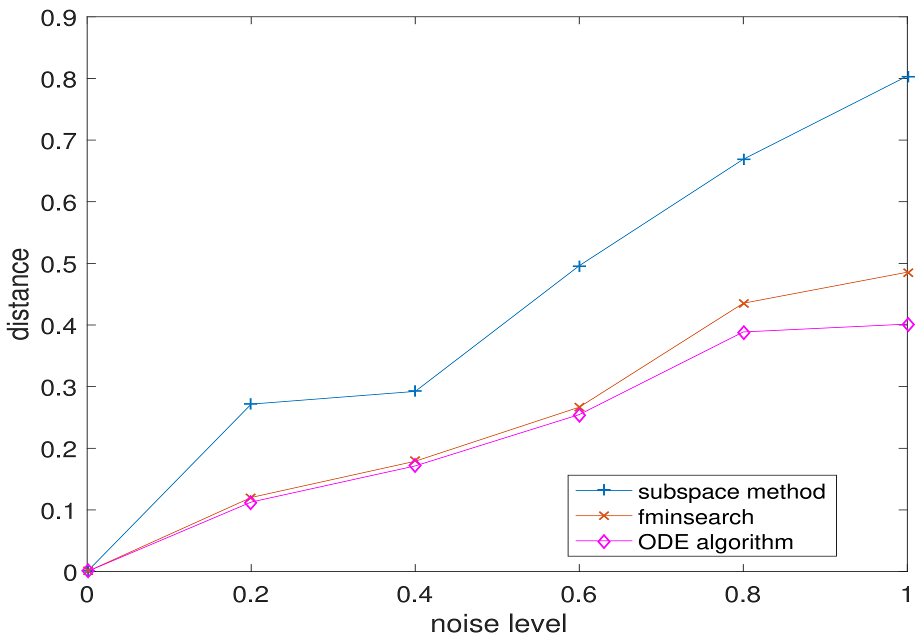

Figure 3: Computation of an Approximate GCD of degree 1 for two 2×2 matrix polynomials of degree 3: distribution of the average distance depending on an increasing level of noise

From the graph we can observe that the proposed ODE-based algorithm obtains better solutions (in terms of accuracy) than the subspace method, as it happened in the case of scalar polynomials [17]. We need to make some comments about the minimization through the Matlab function *fminsearch*. People familiar with Matlab know this function needs an initial approximation in input, so we can ask if the performances observed in Figure 3 depend on the (possibly poor) initialization. In Figure 3 the initial estimate is the solution computed by the subspace method, so it is not a bad choice but neither the best one since we observe the (average) computed distances are bigger than the ones computed by the ODE algorithm. If we initialize the function with the GCD computed by the proposed ODE-based algorithm, the solutions computed by the two approaches become very close, even if the results are not exactly the same due to the issues listed in Section 4.4. In Figure 4 we observe a similar numerical example where we added the distances computed by the function *fminsearch* with different initializations (random, solution of the subspace method, solution of the ODE algorithm). We notice how the different initial estimates for the function *fminsearch* influence the accuracy of the obtained solution.

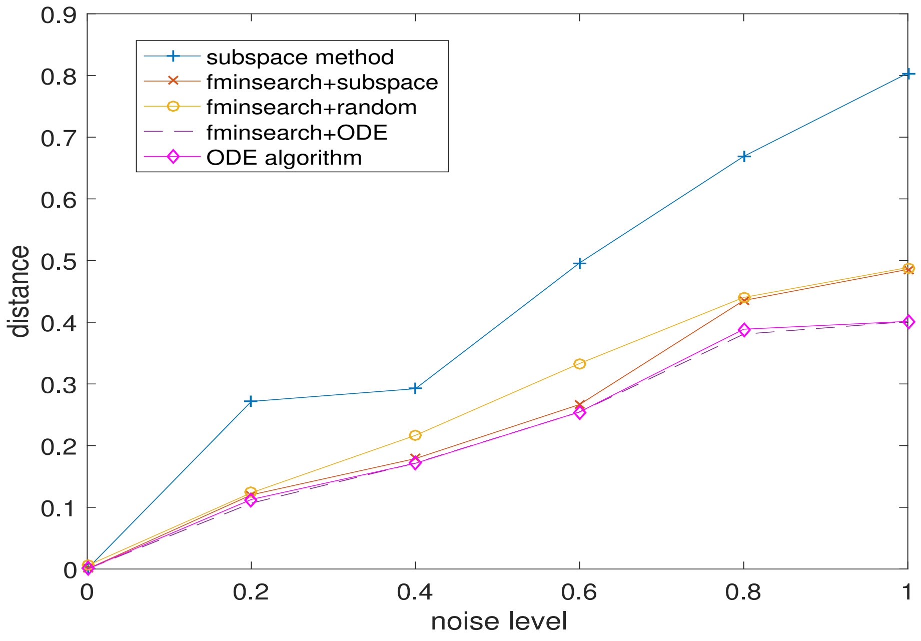

Figure 4: Computation of an Approximate GCD of degree 1 for two 2×2 matrix polynomials of degree 3: distribution of the average distance depending on an increasing level of noise; we initialized the Matlab minimization in different ways

Remark 6. (Computational time) The subspace method is very fast due to its low number of arithmetic operations. The proposed ODE-based algorithm is (on average) faster than the function *fminsearch*, whose performances depend on the initial estimate.

6. Applications in system and control theory

We present in this section an application of the proposed algorithms. It extends the computation of *distance to uncontrollability* from Single Input Single Output systems (presented in [1]) to Multi Input Multi Output systems. However we remind that any problem involving exact GCD computation for matrix polynomials can be seen as an approximate GCD computation problem whenever the coefficient are inexact, e.g. they come from measurements, computations etc.

Controllability for LTI systems. Consider the linear time invariant system $\mathcal{B}$ defined by its state space representation

$$\begin{cases} \dot{x} & = Ax + Bu \\ y & = Cx + Du \end{cases} \tag{16}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$. The classical notion of controllability for (16) is a property of the matrices A, B and it is related to the rank of the matrix

$$\mathcal{C}(A, B) = (B \quad AB \quad \cdots \quad A^{n-1}B). \tag{17}$$

In particular the system (16) is state controllable if and only if the matrix $\mathcal{C}$ in (17) is full rank. This definition of controllability is not a property of the system, but of the matrices A and B; consequently distance problems associated to the matrix $\mathcal{C}$ may not have a well-defined solution since the same system (16) can be represented by different parameters $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ (for example choosing a different basis or considering a bigger state dimension).

In order to avoid these issues we use the behavioral setting [18, 22, 23], where the notion of controllability is a property of the system and not of the parameters we choose for its representation. In this framework, the system (16) is viewed as the set of its trajectories. The controllability property is the possibility of concatenating any two trajectories, up to a delay of time.

Definition 5. *Let $\mathcal{B}$ be a time invariant dynamical system, which is a set of trajectories (vector valued functions of time). $\mathcal{B}$ is said to be controllable if for all $w_1, w_2 \in \mathcal{B}$ there exists a $T > 0$ and a $w \in \mathcal{B}$ such that*

$$w(t) = \begin{cases} w_1(t) & \text{for } t < 0 \\ w_2(t) & \text{for } t \geq T \end{cases}$$

A system is uncontrollable if it is not controllable.

Any linear time invariant system admit a kernel representation [24]; hence given the system $\mathcal{B}$, there is a polynomial matrix $R(z) = (P(z) \; Q(z)) \in \mathbb{R}^{p\times(m+p)}$ such that

$$\mathcal{B}(R) = \{w \mid R_0 w + R_1 \sigma w + \cdots + R_l \sigma^l w = 0\} \tag{18}$$

where σ is the shift operator (in the discrete case). The controllability property is related to the rank of the matrix polynomial $R(z)$, and in particular we have the following Lemma [18]:

Lemma 4. *The system $\mathcal{B}$ is controllable (according to Definition 5) if and only if the polynomial matrix*

$$R(z) = R_0 + R_1 z + \cdots + R_l z^l$$

is left prime, i.e $R(z)$ is full row rank for all z.

Distance to uncontrollability. Alternatively to (18), a Multi Input Multi Output linear time invariant system can be represented by its input/output representation

$$\mathcal{B}_{i/o}(P, Q) = \left\{ \begin{bmatrix} u \\ y \end{bmatrix} \middle| P(\sigma)y = Q(\sigma)u \right\}$$

where we split the vector w in (18) into two blocks (the inputs u and the outputs y) and we partition the matrix $R = (Q \quad -P)$ accordingly. As a consequence of Lemma 4 we have

Corollary 2. *[25] The presence of left common factors in P and Q leads to loss of controllability.*

Let $\mathcal{L}_{uc}$ be the set of uncontrollable linear time invariant systems with $m \geq 1$ inputs and $p \geq 1$ outputs,

$$\mathcal{L}_{uc} = \{\mathcal{B} \mid \mathcal{B} \text{ uncontrollable MIMO LTI system}\}$$

and define the distance between two arbitrary systems by

$$dist(\mathcal{B}(P, Q), \mathcal{B}(\bar{P}, \bar{Q})) = \| \begin{pmatrix} P & Q \end{pmatrix} - \begin{pmatrix} \bar{P} & \bar{Q} \end{pmatrix} \|_F,$$

where the polynomial matrices are identified by a vector whose elements are their coefficients[1]. The problem of computing the distance to uncontrollability is the following:

Problem 2. *Given a controllable system $\mathcal{B}(P, Q)$, find*

$$d(\mathcal{B}) = \min_{\bar{\mathcal{B}} \in \mathcal{L}_{uc}} dist(\mathcal{B}, \bar{\mathcal{B}}).$$

In order to solve the non convex optimization Problem 2, we aim at perturbing the (left) coprime polynomial matrices P and Q in a minimal way till they have a (left) common factor of degree 1. The solution can be computed by the algorithm proposed in Section 4.

[1]The parameters P and Q which identify the system are not unique. In order to have a well posed definition of distance we can assume P to be monic. This involves however some loss of generality.

7. Conclusions

We generalized two algorithms for computing approximate greatest common divisors from scalar to matrix polynomials. The first is a fast and computationally cheap algorithm which extract the informations about the GCD from the resultant, while the second is a more accurate algorithm based on a two level iteration, which looks for the stationary points of a gradient system associated to a suitable functional. We showed how the performances are similar to the scalar case, and we described how to use the algorithms for computing the distance to uncontrollability for a MIMO LTI system.

Acknowledgments. N. G. thanks the Italian INdAM GNCS (Gruppo Nazionale di Calcolo Scientifico) for financial support. I. M. received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC Grant agreement number 258581 "Structured low-rank approximation: Theory, algorithms, and applications" and Fund for Scientific Research Vlaanderen (FWO) projects G028015N "Decoupling multivariate polynomials in nonlinear system identification" and G090117N "Block-oriented nonlinear identification using Volterra series"; and Fonds de la Recherche Scientifique (FNRS) – FWO Vlaanderen under Excellence of Science (EOS) Project no 30468160 "Structured low-rank matrix / tensor approximation: numerical optimization-based algorithms and applications".

References

[1] I. Markovsky, A. Fazzi, N. Guglielmi, Applications of Polynomial Common Factor Computation in Signal Processing, in: Latent Variable Analysis and Signal Separation. Lecture Notes in Computer Science, Springer, 2018, pp. 99–106.

[2] I. Gohberg, P. Lancaster, L. Rodman, Matrix Polynomials, SIAM, 2009.

[3] E. N. Rosenwasser, B. P. Lampe, Multivariable Computer-controlled Systems, Communications and Control Engineering, Springer London, London, 2006.

[4] E. Emre, Nonsingular factors of polynomial matrices and (A, B)-invariant subspaces, SIAM J. Control Optim. 18 (1980) 288–296. doi:10.1137/0318020.

[5] G. D. Forney Jr., Minimal Bases of Rational Vector Spaces, with Applications to Multivariable Linear Systems, SIAM J. Control 13 (1975) 493–520. doi:10.1137/0313029.

[6] J. C. Basilio, B. Kouvaritakis, An algorithm for coprime matrix fraction description using Sylvester matrices, Linear Algebra Its Appl. 266 (1997) 107–125. doi:10.1016/S0024-3795(96)00636-2.

[7] C. C. Mac Duffee, The Theory of Matrices, Chelsea Publishing Company, New York, 1946.

[8] W. A. Wolovich, Linear Multivariable Systems, Vol. 11 of Applied Mathematical Sciences, Springer New York, New York, NY, 1974.

[9] B. Anderson, E. Jury, Generalized Bezoutian and Sylvester matrices in Multivarioable Linear Control, IEEE Trans. Autom. Control 21 (1976) 551–556. doi:10.1109/TAC.1976.1101263.

[10] R. R. Bitmead, S. Y. Kung, B. D. Anderson, T. Kailath, Greatest Common Divisors via Generalized Sylvester and Bezout Matrices, IEEE Trans. Autom. Control 23 (1978) 1043–1047. doi:10.1109/TAC.1978.1101890.

[11] I. Gohberg, G. Heinig, The Resultant Matrix and its Generalizations. I. The Resultant Operator for Matrix Polynomials, in: Convolution Equations and Singular Integral Operators, Birkhäuser Basel, Basel, 2010, pp. 65–88.

[12] I. Gohberg, M. A. Kaashoek, L. Lerer, The resultant for regular matrix polynomials and quasi commutativity, Indiana Univ. Math. J. 57 (2008) 2793–2814. doi:10.1512/iumj.2008.57.3698.

[13] M. A. Kaashoek, L. Lerer, Quasi Commutativity of Regular Matrix Polynomials: Resultant and Bezoutian, in: Topics in Operator Theory, Birkhäuser Basel, Basel, 2010, pp. 297–314.

[14] M. Moness, B. Lantos, Exact Computation of The Greatest Common Divisor of Two Polynomial Matrices, Period. Polytech. Electr. Eng. (Arch.) 26 (1982) 266–280.

[15] W. Qiu, Y. Hua, K. Abed-Meralm, A subspace method for the computation of the GCD of polynomials, Autom. 33 (1997) 741–743.

[16] N. Guglielmi, I. Markovsky, An ODE-Based Method for Computing the Distance of Coprime Polynomials to Common Divisibility, SIAM J. Numer. Anal. 55 (2017) 1456–1482. doi:10.1137/15M1018265.

[17] A. Fazzi, N. Guglielmi, I. Markovsky, An ODE-based method for computing the approximate greatest common divisor of polynomials, Numer. Algorithms (in press). doi:10.1007/s11075-018-0569-0.

[18] J. W. Polderman, J. C. Willems, Introduction to Mathematical Systems Theory, Vol. 26 of Texts in Applied Mathematics, Springer New York, New York, NY, 1998.

[19] J. J. Sylvester, On a theory of the Syzygetic relations of two rational integral functions, comprising an application to the theory of the Sturm's functions, and that of the greatest algebraical common measure, Philos. Trans. 143 (1853) 407–548.

[20] T. Kato, Perturbation theory for linear operators; 2nd ed., Grundlehren Math. Wiss., Springer, Berlin, 1976.

[21] S. Barnett, Polynomials and Linear Control Systems, Marcel Dekker, Inc., New York, NY, USA, 1983.

[22] J. C. Willems, The behavioral approach to open and interconnected systems: Modeling by tearing, zooming, and linking, IEEE Control Syst. Mag. 27 (2007) 46–99.

[23] I. Markovsky, J. C. Willems, S. Van Huffel, B. De Moor, Exact and Approximate Modeling of Linear Systems: A Behavioral Approach, SIAM, 2006.

[24] J. C. Willems, From time series to linear system Part I. Finite dimensional linear time invariant systems, Autom. 22 (1986) 561–580. doi:10.1016/0005-1098(86)90066-X.

[25] J. C. Willems, A framework for the study of dynamical systems, in: Kaashoek M.A., van Schuppen J.H., Ran A.C.M. (eds) Realization and Modelling in System Theory. Progress in Systems and Control Theory, vol 3, Birkhäuser Boston, 1990, pp. 43–59.